Local invariant feature detection

Tues Oct 6
Kristen Grauman
UT Austin

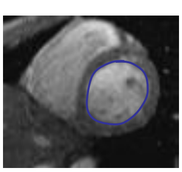
Last time

- Fitting an arbitrary shape with "active" deformable contours

Deformable contours

a.k.a. active contours, snakes

Given: initial contour (model) near desired object
Goal: evolve the contour to fit exact object boundary



Main idea: elastic band is iteratively adjusted so as to

- be near image positions with high gradients, **and**
- satisfy shape "preferences" or contour priors

[Snakes: Active contour models, Kass, Witkin, & Terzopoulos, ICCV1987]

Figure credit: Yuri Boykov

Deformable contours: intuition

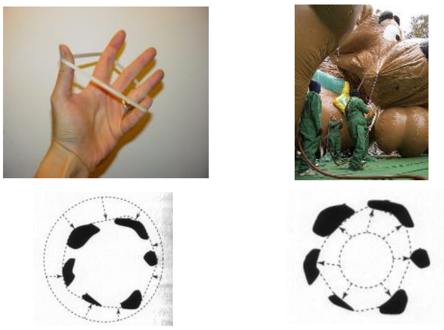
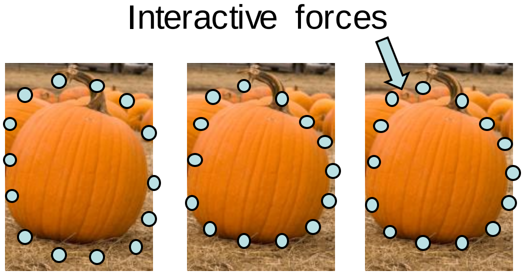


Image from http://www.healthline.com/blogs/exercise_fitness/uploaded_images/HandBand2-795868.JPG

Aspects we need to consider

- Representation of the contours
- Defining the energy functions
  - External
  - Internal
- Minimizing the energy function
- Extensions:
  - Tracking
  - Interactive segmentation

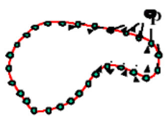
Interactive forces



How can we implement such an *interactive* force with deformable contours?

Interactive forces

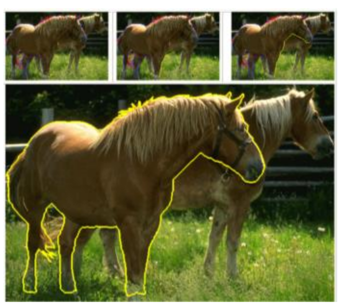
- An energy function can be altered online based on user input – use the cursor to push or pull the initial snake away from a point.
- Modify external energy term to include:



$$E_{push} = \sum_{i=0}^{n-1} \frac{r^2}{|v_i - p|^2}$$

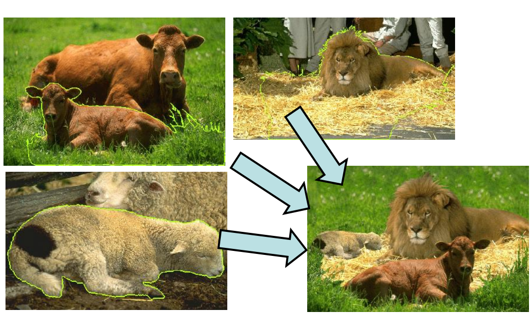
Nearby points get pushed hardest

Intelligent scissors



- http://rivit.cs.byu.edu/Eric/Eric.html

Intelligent scissors



- http://rivit.cs.byu.edu/Eric/Eric.html

Intelligent scissors

Another form of interactive segmentation:

Compute optimal paths **from every point to the seed** based on edge-related costs.

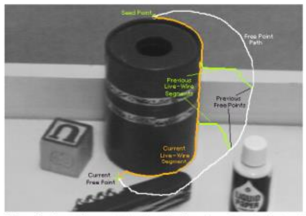


Figure 2: *Image demonstrating how the live-wire segment adapts and snaps to an object boundary as the free point moves (via cursor movement). The path of the free point is shown in white. Live-wire segments from previous free point positions (t_0, t_1, and t_2) are shown in green.*

VIDEO

[Mortensen & Barrett, SIGGRAPH 1995, CVPR 1999]

Deformable contours: pros and cons

Pros:

- Useful to track and fit non-rigid shapes
- Contour remains connected
- Possible to fill in "subjective" contours
- Flexibility in how energy function is defined, weighted.

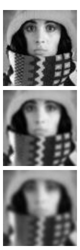
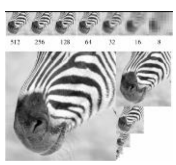
Cons:

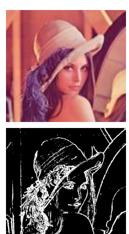
- Must have decent initialization near true boundary, may get stuck in local minimum
- Parameters of energy function must be set well based on prior information

Recap: Deformable contours

- Deformable shapes and active contours are useful for
  - Segmentation: fit or "snap" to boundary in image
  - Tracking: previous frame's estimate serves to initialize the next
- Fitting active contours:
  - Define terms to encourage certain shapes, smoothness, low curvature, push/pulls, …
  - Use weights to control relative influence of each component cost
  - Can optimize 2d snakes with Viterbi algorithm.
- Image structure (esp. gradients) can act as attraction force for *interactive* segmentation methods.

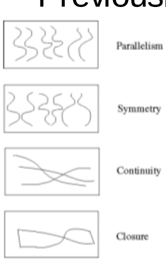
Previously: Features and filters

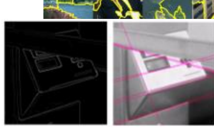



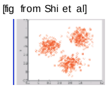
Transforming and describing images; textures, colors, edges

Previously: Grouping & fitting



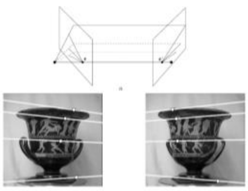



[fig from Shi et al]

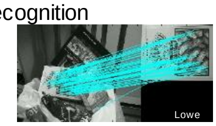


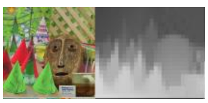
Clustering, segmentation, fitting; what parts belong together?

Now: Multiple views


Hartley and Zisserman

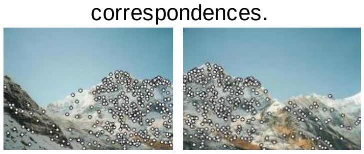
Matching, invariant features, stereo vision, instance recognition





Now: Local features

Multi-view matching relies on **local feature** correspondences.



How to detect *which local features* to match?

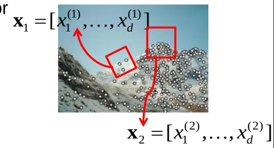
Detecting local invariant features

- Detection of interest points
  - Harris corner detection
  - Scale invariant blob detection: LoG
- (Next time: description of local patches)

Local features: main components

1) Detection: Identify the interest points

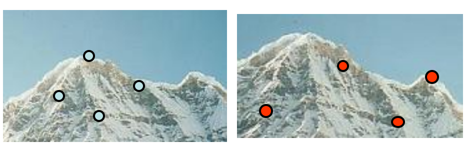
2) Description: Extract vector feature descriptor surrounding each interest point.



3) Matching: Determine correspondence between descriptors in two views

Goal: interest operator repeatability

- We want to detect (at least some of) the same points in both images.

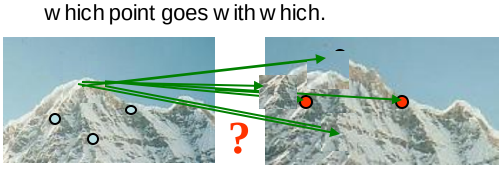


No chance to find true matches!

- Yet we have to be able to run the detection procedure *independently* per image.

Goal: descriptor distinctiveness

- We want to be able to reliably determine which point goes with which.



- Must provide some invariance to geometric and photometric differences between the two views.

Local features: main components

1) Detection: Identify the interest points

2) Description: Extract vector feature descriptor surrounding each interest point.

3) Matching: Determine correspondence between descriptors in two views

- What points would you choose?

Detecting corners

Detecting corners

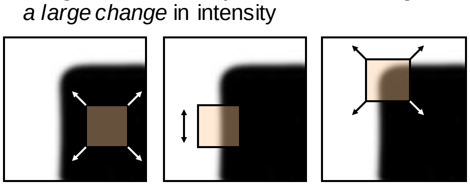
Compute "cornerness" response at every pixel.

Detecting corners

Corners as distinctive interest points

We should easily recognize the point by looking through a small window

Shifting a window in *any direction* should give *a large change* in intensity



"flat" region: no change in all directions

"edge": no change along the edge direction

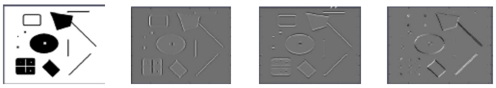
"corner": significant change in all directions

Slide credit: Alyosha Efros, Darya Frolova, Denis Simakov

Corners as distinctive interest points

$$M = \sum w(x,y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix}$$

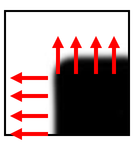
2 x 2 matrix of image derivatives (averaged in neighborhood of a point).



Notation: $I_x \Leftrightarrow \frac{\partial I}{\partial x}$ $I_y \Leftrightarrow \frac{\partial I}{\partial y}$ $I_x I_y \Leftrightarrow \frac{\partial I}{\partial x}\frac{\partial I}{\partial y}$

What does this matrix reveal?

First, consider an axis-aligned corner:



What does this matrix reveal?

First, consider an axis-aligned corner:

$$M = \sum \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

This means dominant gradient directions align with x or y axis

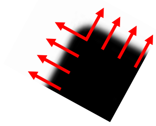
Look for locations where **both** λ's are large.

If either λ is close to 0, then this is **not** corner-like.

What if we have a corner that is not aligned with the image axes?

What does this matrix reveal?

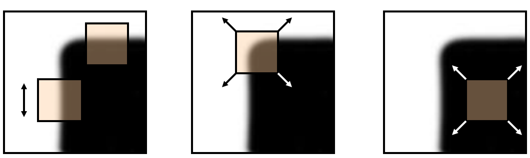
Since *M* is symmetric, we have $M = X \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} X^T$



$Mx_i = \lambda_i x_i$

The *eigenvalues* of *M* reveal the amount of intensity change in the two principal orthogonal gradient directions in the window.

Corner response function



"edge":
$\lambda_1 >> \lambda_2$
$\lambda_2 >> \lambda_1$

"corner":
λ_1 and λ_2 are large,
$\lambda_1 \sim \lambda_2$;

"flat" region
λ_1 and λ_2 are small;

$$f = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2}$$

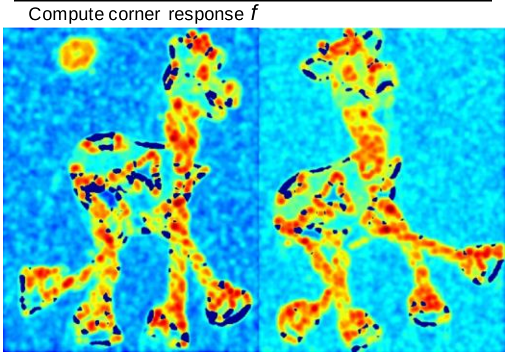
Harris corner detector

1) Compute *M* matrix for each image window to get their *cornerness* scores.
2) Find points whose surrounding window gave large corner response (*f*> threshold)
3) Take the points of local maxima, i.e., perform non-maximum suppression

Harris Detector: Steps

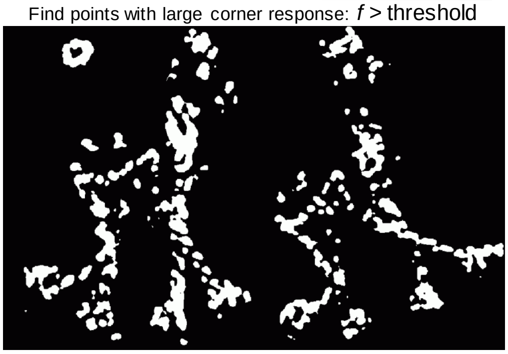
Harris Detector: Steps

Compute corner response *f*



Harris Detector: Steps

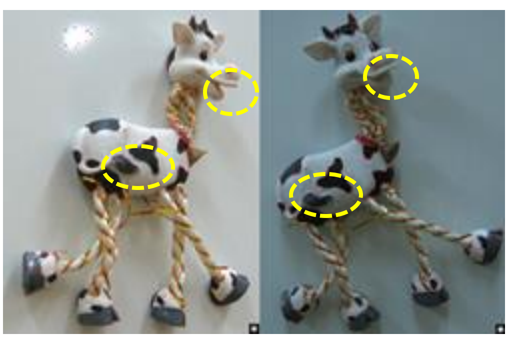
Find points with large corner response: f > threshold



Harris Detector: Steps

Take only the points of local maxima of *f*

Harris Detector: Steps



Properties of the Harris corner detector

Rotation invariant? Yes

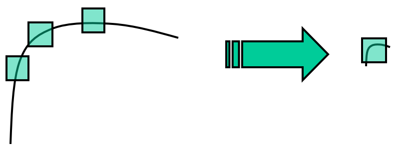
$$M = X\begin{bmatrix}\lambda_1 & 0 \\ 0 & \lambda_2\end{bmatrix}X^T$$

Scale invariant?

Properties of the Harris corner detector

Rotation invariant? Yes

Scale invariant? No

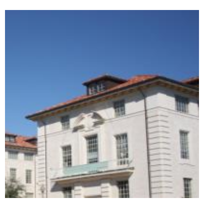


All points will be classified as edges

Corner !

Scale invariant interest points

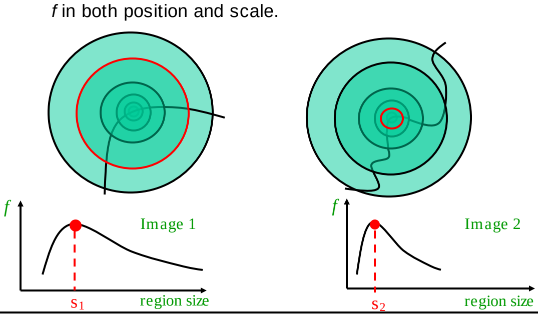
How can we independently select interest points in each image, such that the detections are repeatable across different scales?



Automatic scale selection

Intuition:

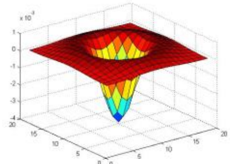
- Find scale that gives local maxima of some function f in both position and scale.

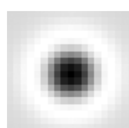


What can be the “signature” function?

Blob detection in 2D

Laplacian of Gaussian: Circularly symmetric operator for blob detection in 2D

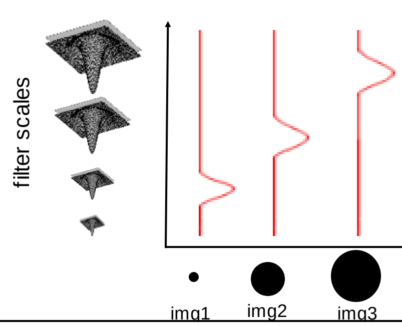




$$\nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2}$$

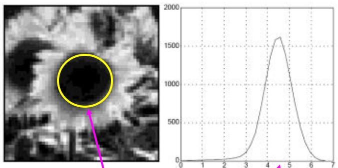
Blob detection in 2D: scale selection

Laplacian-of-Gaussian = “blob” detector $\nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2}$



Blob detection in 2D

We define the *characteristic scale* as the scale that produces peak of Laplacian response



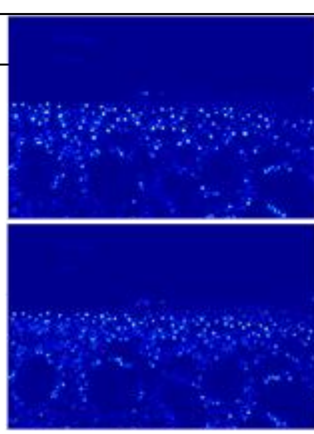
characteristic scale

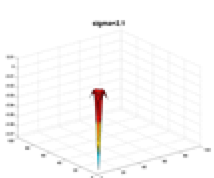
Slide credit: Lana Lazebni

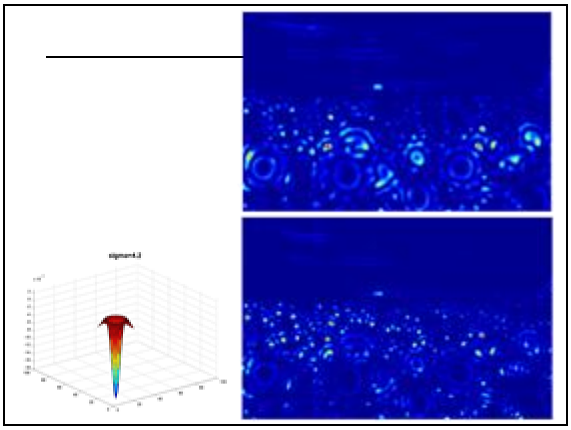
Example

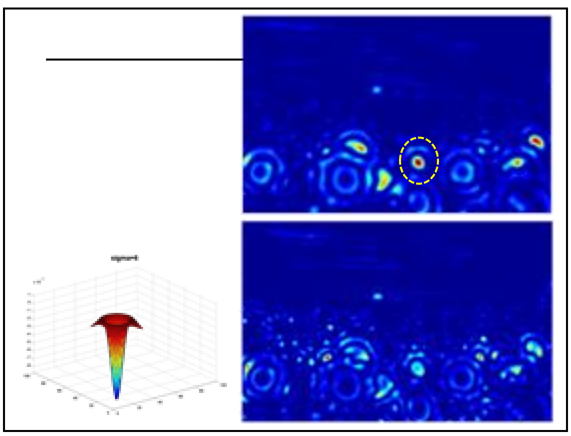
Original image at ¾ the size

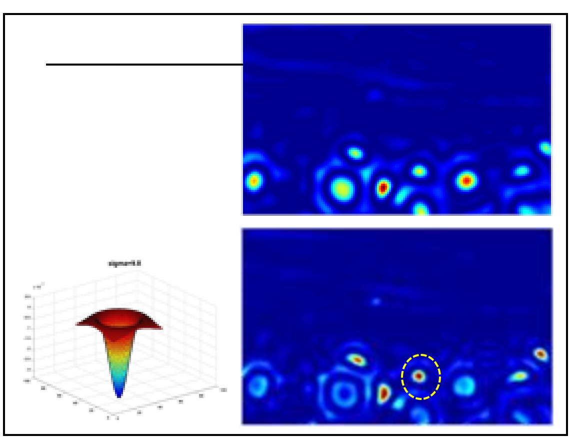
Original image at ¾ the size

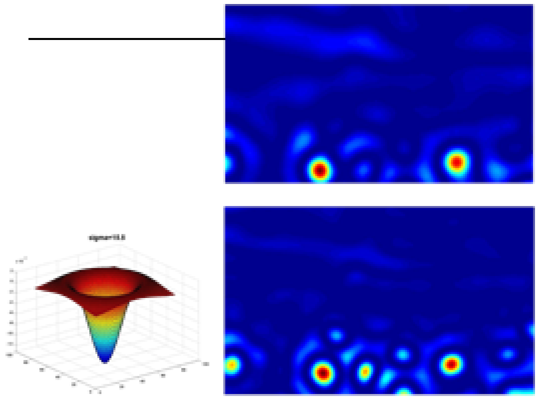


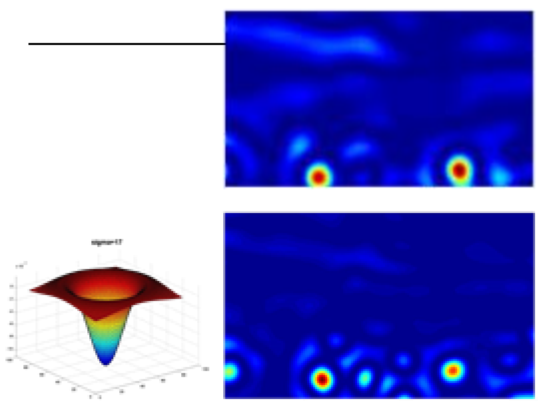






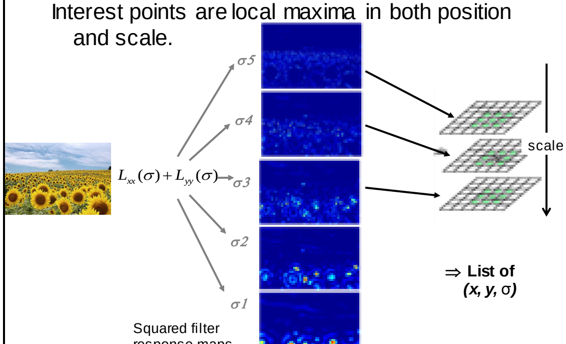






Scale invariant interest points

Interest points are local maxima in both position and scale.



Scale-space blob detector: Example

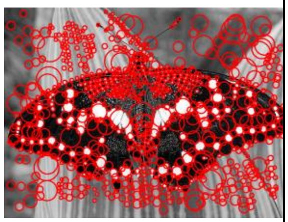



Image credit: Lana Lazebnik

Summary

- Interest point detection
  - Harris corner detector
  - Laplacian of Gaussian, automatic scale selection